The Report Committee for Xoab Perez

certifies that this is the approved version of the following report:

Model Selection for Tumor Growth with Nonlinear Mechanical Effects

APPROVED BY

SUPERVISING COMMITTEE:

Thomas Yankeelov, Supervisor

Danial Faghihi-Shahrestani

Model Selection for Tumor Growth with Nonlinear Mechanical Effects

by

Xoab Perez

REPORT

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Computational Science, Engineering, and Mathematics

The University of Texas at Austin

May 2019

Dedicated to Mom and Beka

Acknowledgments

It has been a privilege to work under the tutelage of some wonderful people here at ICES. I would like to thank Danial Faghihi and Xinzeng Feng in particular. Their enthusiasm, humility, and deep commitment to their work was a great source of inspiration. I can only hope to emulate their selfless generosity and support.

Model Selection for Tumor Growth with Nonlinear Mechanical Effects

Xoab Perez, M.S.C.S.E.M.
The University of Texas at Austin, 2019

Supervisor: Thomas Yankeelov

Accurately modeling *in vivo* tumor growth is a persistent challenge due to the complexity of tumors and their environments. Accurate models are sought after for their potential to guide treatments and help researchers discover or better understand the underlying biological processes. Previous research has identified a reaction-diffusion formulation coupled with mechanical forces that performs well at modeling tumor growth. The focus of the current research was a similar formulation with a nonlinear constitutive equation instead of a linear constitutive equation for the mechanical forces. In this study the models performed similarly, with the nonlinear model predictions being slightly closer to the actual actual tumor growth on average. This indicates that the linear model may be a sufficiently close approximation, though the parameter estimation procedure needs improvement. Then other nonlinear models can be studied easily using the code developed for this work.

Contents

Chapter 1 Introduction **1**

Chapter 2 Methods and Mathematical Formulations **4**

2.1 Murine glioma data 4

2.2 Models of tumor growth 4

2.2.1 Reaction diffusion system 5

2.2.2 Mechanics: infinitesimal strain theory formulation (ISTF) . . . 6

2.2.3 Mechanics: finite strain theory formulation (FSTF) 8

2.2.4 Reaction-diffusion with mechanical coupling 10

2.2.5 Discrete formulation 10

2.3 Parameter estimation and model comparison 13

2.3.1 Validation 14

Chapter 3 Results **15**

3.1 Experimental setup 15

3.2 Parameter estimation 16

3.3 Model comparison 18

Chapter 4 Discussion **26**
4.1 Summary of key results 26
4.2 Study limitations 27
4.2.1 Models 27
4.2.2 Parameter estimation 27

Chapter 5 Conclusion **29**

Appendix A FEniCS Implementation **30**

Appendix B On Bayesian Methods for Model Calibration and Selection . **39**
B.1 Model Calibration 39
B.1.1 Ordinary Least Squares (OLS) 40
B.1.2 Bayesian Inference 40
B.1.3 Algorithms for sampling a posterior distribution 43
B.2 Model selection 53
B.2.1 Hierarchical models 53
B.2.2 Model comparison 54
B.2.3 Model Complexity 56
B.3 Examples of Calibration and Selection 56
B.3.1 OLS Solutions 58
B.3.2 Bayesian Inference Solutions 59
B.3.3 Model Comparison 63

Bibliography **65**

Chapter 1

Introduction

Cancer is a complex set of diseases which have occurred in humans for thousands of years, with recordings of cancerous tumors existing from the time of ancient Egyptians [28]. Treatment options have broadened since then and now include not only surgery, but also chemotherapy, radiation, immunotherapy, targeted therapies, and hormonal therapies. Although these can be very effective at killing cancer cells or blocking tumors from growing and metastasizing, the complexity of cancer makes complete elimination difficult, and side effects can be severe and drastically reduce patient quality of life. Other problems with treatment include difficulty diagnosing the cancer correctly to begin with, determining the optimal treatment plan, and drug resistance. Discovering new treatment options and calculating proper dosages is also challenging. Current methods require a large patient population for testing, and both inter- and intra-patient variance cause large differences in the treatment outcomes.

Mathematical modeling of cancer may be able to provide important improvements to the issues described above. Mathematical models can describe the

evolution of cancer and, through simulations of these models, enable the computational study of effects on the modeled environment from treatments. In this way, estimates of tumor growth can be made directly and a myriad of treatment options can be systematically and quantitatively compared *in silico*. Optimization of treatments can also be done to reduce side effects while maximizing efficacy. Fewer patients will then be needed for clinical trials, and these trials can then be optimized after determining through simulations which parameters should be the focus. Patients can then benefit from knowledge acquired through computational simulations tailored to their particular biology.

Much work is being done towards developing mathematical models that accurately describe tumor growth in the form of agent-based models [16, 29], continuum models [13], and hybrid models [27, 4]. One such model of tumor growth is Fisher's equation [8], a reaction-diffusion system with logistic growth for tumor cells. This model describes cancerous cells proliferating and diffusing throughout the local tissue, with growth being capped by some physical or biological limit, as seen in [18]. In reality, the dispersion is not completely unchecked, and experiments have shown a reduction in growth when tumor cells encounter physical resistance [12]. The standard reaction-diffusion model has been extended to include this by using solid mechanics and linking the gradient of tumor cells to solid stresses [13]. The diffusion coefficient is reduced by some factor in areas of high stress, mitigating growth in these directions. In our laboratory's model, the stress-strain relationship is described by a linear elastic constitutive model, a simplifying assumption which is typical for infinitesimal strain. However, tumors are known

to grow many times their initial size, which causes strains that are large by any strain measure. Under large strains, materials exhibit nonlinear behavior, either softening like some metals or, as is often the case with biological tissues, stiffening [5, 15]. Thus, I propose that a nonlinear constitutive model of stress-strain would improve the tumor growth model.

The overall goal of this project is to test whether using a nonlinear constitutive model that incorporates large strains will enable the tumor growth model to better fit data for murine glioma growths. We will consider this to be achieved if the sum of the pixel-wise differences between the true and modeled tumor is less when modeling with the nonlinear constitutive model (i.e., if the L^2 norm of the error is smaller). In the following chapter, the mathematical formulations will be presented along with the data used and the procedure for calculating the error.

Chapter 2

Methods and Mathematical Formulations

2.1 Murine glioma data

Magnetic resonance imaging data was acquired during tumor development in previous experiments Hormuth et al. [14]. Six rats were injected with C6 glioma cells in the cerebrum and subsequently imaged periodically. Tumor cell counts were estimated per voxel on a grid of size 41x61x16. In order to reduce computational time, a 2D formulation was developed (described in section 2.2.2) which required data from the center of the tumor. Thus, a 41x61 grid from each rat's data was used. An example of the 2D data used for parameter estimation and model comparison is shown in figure 2.1.

2.2 Models of tumor growth

This research focused on modeling the tumor cells as a homogeneous group within rat brains, a simplification of the reality in which there are multiple can-

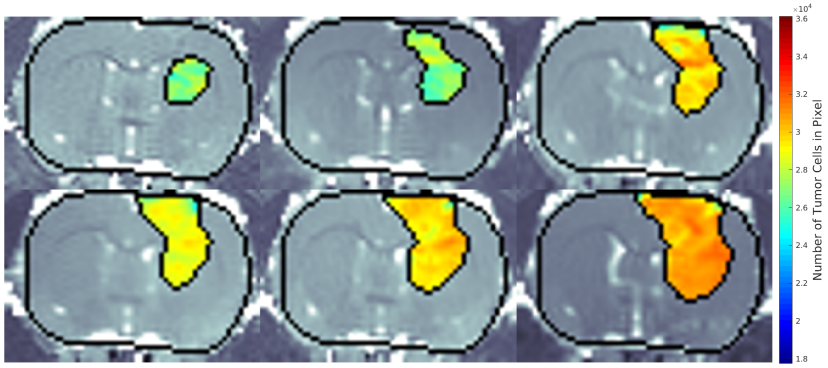

Figure 2.1: A brain slice showing progression of tumor growth starting from top left to top right, then bottom left to bottom right.

cer cell phenotypes making a heterogeneous tumor. The behavior of these cells is generic and follows documented observations - multiplying, dispersing, and responding to certain stimuli within the domain, which in this case is the stresses from expansion. Cell death is not accounted for, and the local environment, i.e. blood vessels and specific tissues, is not explicitly modeled. Instead, the environmental influences are assummed to be implicitly accounted for in specific model parameters which are estimated (see section 2.3.) The following sections provide specific details on the implementation.

2.2.1 Reaction diffusion system

A reaction-diffusion (RD) system is capable of describing some of the behaviors above and is thus an apt starting point. Let $\phi(\mathbf{x}, t)$ be the concentration of tumor cells at location $\mathbf{x} \in \Omega \subset \mathbb{R}^2$, where Ω is a 2D domain, and time $t \in [0, T]$.

Then, the partial differential equation in the following set of equations describes the rate of change of tumor cells over time:

$$\begin{cases} \frac{\partial \phi(\mathbf{x},t)}{\partial t} &= D\Delta\phi(\mathbf{x},t) + k(\mathbf{x})\phi(\mathbf{x},t)\left(1 - \frac{\phi(\mathbf{x},t)}{\theta}\right) & \text{in} \quad \Omega \times (0,T) \\ \phi(\mathbf{x},0) &= \phi_{t,0} & \text{in} \quad \Omega \\ \frac{\partial \phi}{\partial \mathbf{n}} &= 0 & \text{on} \quad \Gamma \times (0,T) \end{cases} \tag{2.1}$$

where $\mathbf{n}$ is the vector normal to the boundary Γ. The concentration diffuses according to parameter D, a diffusion field, and grows according to parameter k, the growth coefficient field, until being limited either by physical or biological constraints which can be accounted for in the carrying capacity parameter θ. The values of these and other parameters are discussed in section 2.3.

To solve this PDE, an initial condition is required. The initial condition used here is the concentration at the beginning of the simulation, taken from the rat data described in section 2.1. A Neumann boundary condition is used so that there is zero flux where the tumor meets the skull.

2.2.2 Mechanics: infinitesimal strain theory formulation (ISTF)

For the assumption of small strains, i.e. when the magnitude of the tumor's displacement gradient $||\nabla \boldsymbol{u}||$ is small, the tumor's infinitesimal strain tensor has only the following linear terms:

$$\boldsymbol{E} = \frac{1}{2}(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T)$$

To model the effect of growth on the mechanical behavior of tumors, it is considered here that the strain tensor is decomposed into deformation $\boldsymbol{E}^S$ and growth $\boldsymbol{E}^G$:

$$\boldsymbol{E} = \boldsymbol{E}^S + \boldsymbol{E}^G$$

For the isotropic growth, we consider here a growth stretch as a function of tumor:

$$\boldsymbol{E}^G = \beta\phi\boldsymbol{I}$$

The Cauchy stress tensor is:

$$\boldsymbol{T} = \boldsymbol{C}(\boldsymbol{E} - \boldsymbol{E}^G) = \boldsymbol{C}\boldsymbol{E}^S$$

where $\boldsymbol{C}$ is the 4th order elasticity tensor. Under linear and isotropic assumptions:

$$\begin{aligned}\boldsymbol{T} &= 2\mu\boldsymbol{E}^S + \frac{2\mu}{1-2\nu}(\mathrm{tr}\boldsymbol{E}^S)\boldsymbol{I} \\ &= 2\mu(\boldsymbol{E} - \beta\phi\boldsymbol{I}) + \frac{2\mu}{1-2\nu}\boldsymbol{I}(\mathrm{tr}\boldsymbol{E} - \mathrm{tr}\boldsymbol{E}^G)\end{aligned}$$

where μ is the bulk shear modulus, ν is Poisson's ratio, and $\mathrm{tr}\boldsymbol{E}^G = 2\beta\phi$ for a 2D formulation. Thus, the equilibrium condition and the stress tensor are as follows, with no body forces or traction forces assumed:

$$\begin{aligned}\nabla \cdot \boldsymbol{T} &= 0 \\ \boldsymbol{T} &= 2\mu(\boldsymbol{E} - \beta\phi\boldsymbol{I}) + \frac{2\mu}{1-2\nu}(\mathrm{tr}\boldsymbol{E} - 2\beta\phi)\boldsymbol{I}\end{aligned}$$

To have a 2-dimensional problem, a plane strain formulation was assumed. This means that the strain in one direction is negligible compared to the strains in the other two directions. With data from a slice through the center of the tumor, this is a reasonable approximation as the tumor will grow outward and not in/out of the slice where there are already tumor cells.

2.2.3 Mechanics: finite strain theory formulation (FSTF)

Let $\boldsymbol{x} = \chi(\boldsymbol{X}, t)$ be the current (deformed) configuration, a function of the reference (undeformed) configuration $\boldsymbol{X}$ and time. The deformation gradient is:

$$\boldsymbol{F} = \frac{\partial \chi}{\partial \boldsymbol{X}}$$

To model the effect of growth on the mechanical behavior of tumors, it is considered here that the deformation gradient is decomposed into deformation $\boldsymbol{F}^S$ and growth $\boldsymbol{F}^G$:

$$\boldsymbol{F} = \boldsymbol{F}^S \boldsymbol{F}^G$$

where $\boldsymbol{F}^G$ is the growth tensor, which we consider here to be a function of the tumor concentration as follows,

$$\boldsymbol{F}^G = \lambda^G \boldsymbol{I} = (\beta\phi + 1)\boldsymbol{I}$$

A compressible neo-Hookean model was chosen to describe the constitutive behavior of the tumor. Many biological tissues exhibit behavior consistent with hy-

perelastic material models like neo-Hookean, and this one was chosen for having only two parameters. In this case the parameters are the same as those required for the linear elastic model. The neo-Hookean model can be described using a strain energy function using invariants of the right Cauchy-Green deformation tensor,

$$
\begin{aligned}
I_{C_1}^S &= \mathrm{tr}(\boldsymbol{C}^S) \\
J^S &= \sqrt{\det(\boldsymbol{C}^S)} = \det(\boldsymbol{F}^S) \\
W &= \frac{\mu}{2}(I_{C_1}^S - 3) + \frac{K}{2}(J^S - 1)^2
\end{aligned}
$$

The first Piola-Kirchhoff stress is used in the equation of equilibrium, and is derived as follows:

$$
\begin{aligned}
\boldsymbol{T} &= \frac{\partial W}{\partial \boldsymbol{F}} = \frac{\partial W}{\partial \boldsymbol{F}^S}\frac{\partial \boldsymbol{F}^S}{\partial \boldsymbol{F}} \\
\frac{\partial W}{\partial \boldsymbol{F}^S} &= \frac{\mu}{(J^S)^{5/3}}(\boldsymbol{B}^S - \frac{1}{3}\mathrm{tr}(\boldsymbol{B}^S)\boldsymbol{I}) + K(J^S - 1)\boldsymbol{I} \\
\frac{\partial \boldsymbol{F}^S}{\partial \boldsymbol{F}} &= \boldsymbol{I} \otimes (\boldsymbol{F}^G)^{-1} = \frac{1}{\lambda^G}
\end{aligned}
$$

The equilibrium formulation for neo-Hookean hyperelasticity is thus:

$$
\begin{aligned}
\nabla \cdot \boldsymbol{T} &= 0 \\
\boldsymbol{T} &= \frac{1}{1+\beta\phi}\left[\frac{\mu}{(J^S)^{5/3}}(\boldsymbol{B}^S - \frac{1}{3}\mathrm{tr}(\boldsymbol{B}^S\boldsymbol{I}) + K(J^S - 1)\boldsymbol{I}\right]
\end{aligned}
$$

2.2.4 Reaction-diffusion with mechanical coupling

There are several possibilities for integrating the mechanical forces with the reaction-diffusion equation. A reasonable and simple such possibility is to have the diffusion be a function of the stress field [7, 20]. In this way, the model follows observed behavior of tumor cell count stagnating upon encountering high stresses. Implementation is thus:

$$D = D_0 \exp(-\gamma_D \sigma_{vm})$$

where σ_{vm} is the Von Mises stress criterion, a measure of total stress at a location given the normal and shear stresses.

2.2.5 Discrete formulation

The open-source software library FEniCS and associated components [3, 23, 22, 21, 24, 1, 17, 2] was chosen to solve the RD system computationally. This library can take meshes and solves a nonlinear variational problem by making use of the formulation. Additionally, it can be used with another software package, dolfin-adjoint, for parameter estimation.

The discretization in time was conducted before forming the variational problem as input to FEniCS, and the formulation was applied to a mesh similar to the one shown in figure 2.2. In this case, backward Euler was chosen for sim-

plicity and its stability properties:

$$\frac{\phi(\mathbf{x}, t_{n+1}) - \phi(\mathbf{x}, t_n)}{\delta t} = D\Delta\phi(\mathbf{x}, t_{n+1}) + k(\mathbf{x})\phi(\mathbf{x}, t_{n+1})\left(1 - \frac{\phi(\mathbf{x}, t_{n+1})}{\theta}\right)$$

where δt is the time step and t_n is the nth time point. After multiplying by a test function v and integrating by parts, the form is:

$$\begin{aligned}\int_\Omega \frac{\phi(\mathbf{x}, t_{n+1}) - \phi(\mathbf{x}, t_n)}{\delta t} v \, d\Omega = &\int_\Omega -D(\nabla\phi(\mathbf{x}, t_{n+1}) \cdot \nabla v) \, d\Omega + \\ &\int_\Omega k(\mathbf{x})\phi(\mathbf{x}, t_{n+1})\left(1 - \frac{\phi(\mathbf{x}, t_{n+1})}{\theta}\right) v \, d\Omega + \\ &\int_\Gamma \frac{\partial\phi(\mathbf{x}, t_{n+1})}{\partial\mathbf{n}} \cdot v \, d\Gamma\end{aligned}$$

The Neumann boundary condition eliminates the final integral, and since the variational form is nonlinear, it can be set up in the following way: find $\phi \in \Phi$, some suitable function space, such that:

$$F(\phi; v) = 0, \quad \forall v \in \hat{V}$$

$$\begin{aligned}F = \int_\Omega \Bigg[&\frac{\phi(\mathbf{x}, t_{n+1}) - \phi(\mathbf{x}, t_n)}{\delta t} v + D(\nabla\phi(\mathbf{x}, t_{n+1}) \cdot \nabla v) \\ &-k(\mathbf{x})\phi(\mathbf{x}, t_{n+1})\left(1 - \frac{\phi(\mathbf{x}, t_{n+1})}{\theta}\right) v \Bigg] \, d\Omega\end{aligned}$$

This form can be solved by FEniCS. See Appendix A for implementation details.

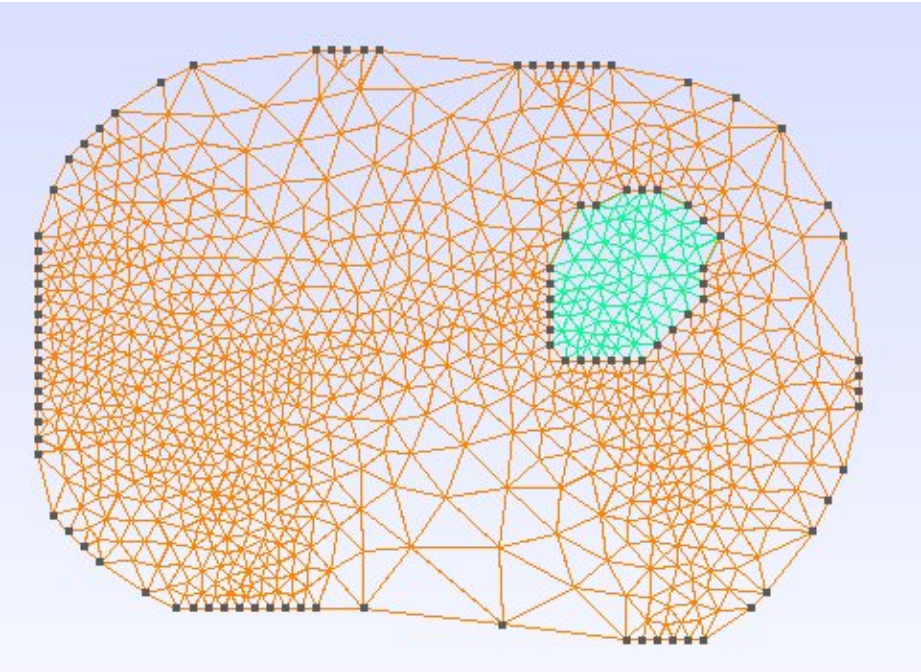

Figure 2.2: An example of a finite element mesh used for this problem.

2.3 Parameter estimation and model comparison

Given multiple data points and a model with unknown parameters or coefficients, there are several ways to estimate the parameter values that allow the model to best fit the data. Parameters can be estimated either deterministically, i.e. each parameter having a specific value, or probabilistically, i.e. each parameter can be a random variable with a distribution. After estimating parameters for different models, there are then many ways to compare the models. In this research, deterministic parameter estimation was used and the models were compared based on their L^2 error norms. An example of probabilistic parameter estimation and model selection is described in Appendix B for future use.

Dolfin-adjoint is a package used for optimization that combines an adjoint solver library, pyadjoint, with FEniCS. After providing a functional $\mathcal{J}$ to minimize, dolfin-adjoint derives the adjoint and tangent linear models for the minimization procedure. The functional to be minimized is the L^2-norm at time t:

$$\mathcal{J}(t) = ||\phi(t) - \phi_{true}(t)||^2_\Omega$$

Regularization was not included but is discussed in chapter 4. There are multiple unknown or uncertain parameters to either be assumed or estimated. Hormuth et al. [14] conducted similar research for a linear elastic mechanically coupled formulation and used the following parameters (differences in this study are mentioned in parentheses):

- θ, assumed to be a constant throughout the domain, was evaluated from the

voxel dimensions of the brain slice images and an assumed packing density to be 50,970.

- k was estimated at each voxel.
- D_0 was estimated as a global constant.
- μ came from the literature and was assigned by region of the brain (assumed to be globally constant at 0.42 kPa).
- ν was set to 0.45, assuming that tissue is nearly incompressible.
- β was set to 1 (estimated and studied at different values)
- γ_D was varied manually to study its effects (estimated)

2.3.1 Validation

Tests were run to check whether the parameter estimation procedure returned valid results. The forward model was run using initial data from one rat specimen along with chosen parameters. The resulting data at two additional time points was used for the parameter estimation. Results were compared to the actual parameters. See chapter 3 for results.

Chapter 3

Results

3.1 Experimental setup

For each rat, imaging data was available at several time points. The first time point was used for the initial condition and the two subsequent time points were used for parameter estimation. Three points were used to capture nonlinearity. The cost function was the difference between the true data and the modeled data at the second and third time point:

$$\min_{D_0,k,\gamma_D,\beta} ||\phi_{true}(x,2) - p(x,2)||^2 + ||\phi_{true}(x,3) - p(x,3)||^2$$

The estimated parameters were then used to run the forward model and compare the output from each model to the actual data at each time point. Since the Dolfin nonlinear solver initially had some trouble converging during the parameter estimation, bounds for the parameters were approximated by running the forward model with various combinations of the parameters and studying when

convergence failed. This led to the following parameter domains for parameter estimation:

$$k \in [0, 10]$$
$$D_0 \in [0, 5]$$
$$\gamma_D \in [.01, 5]$$
$$\beta \in [0.01, 10]$$

3.2 Parameter estimation

Validation testing with simulated data worked well when only estimating D_0 and k (results not shown here), but upon introduction of γ_D and β, the procedure was not as effective at capturing the true parameters. See table 3.1 and figure 3.1 for the prescribed and estimated values. See figure 3.2 for the simulated and predicted tumor growth.

	True	ISTF	FSTF
D_0	1.0	.92	2.92
γ_D	.1	.01	3.82
β	.3	5.2	2.58

Table 3.1: Validation of parameter estimation procedure.

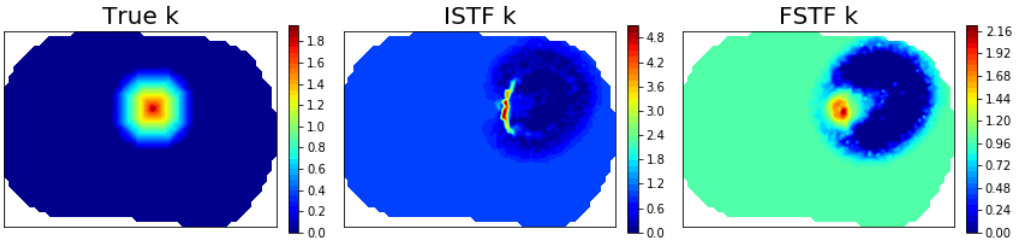

Figure 3.1: True and estimated k for validation.

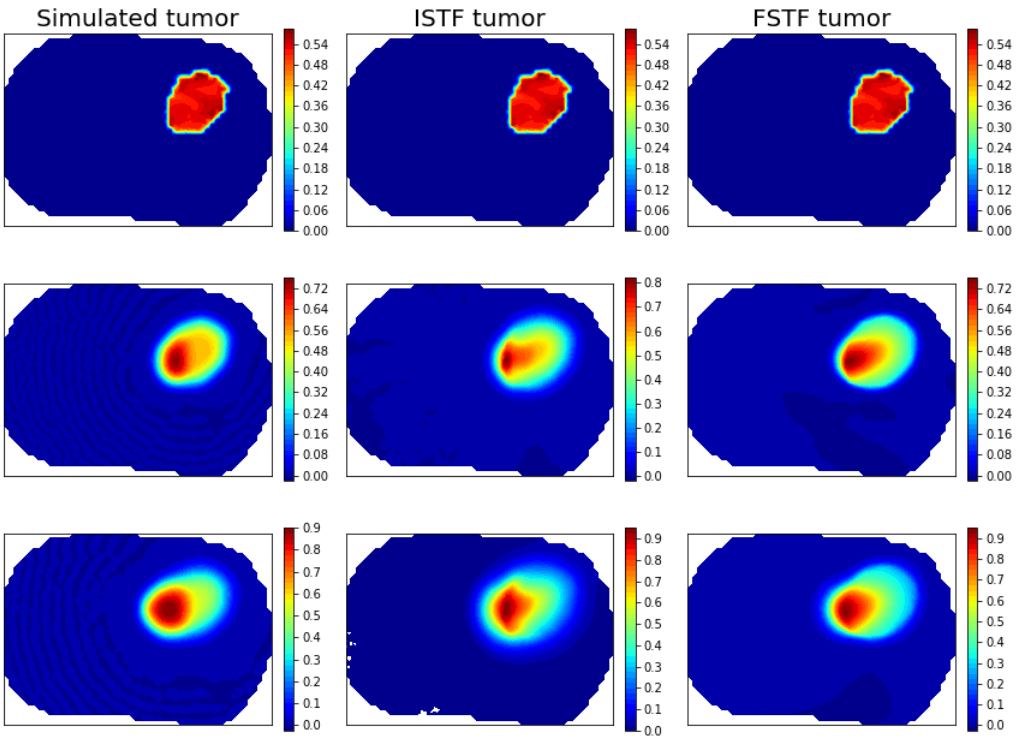

Figure 3.2: True and estimated tumor growth for validation at 3 time points.

The parameter estimation procedure was then followed for different specimens, and a summary of parameters is shown in table 3.2.

	Rat 1		Rat 5		Rat 6		Rat 9	
	ISTF	FSTF	ISTF	FSTF	ISTF	FSTF	ISTF	FSTF
D_0	1.45	1.11	4.81	1.04	1.02	0.99	1.90	1.46
γ_D	1.73	1.87	2.31	1.21	1.13	1.24	0.93	0.30
β	1.73	2.44	2.31	1.00	1.13	2.39	0.93	1.59

Table 3.2: Parameter estimation summary.

3.3 Model comparison

The data misfit (using the functional $\mathcal{J}$ described above) is shown in table 3.3. Figures 3.3, 3.4, 3.5, and 3.6 show the true tumor data superimposed with the outlines of the tumor data modeled assuming linear elasticity and hyperelasticity, respectively. Figures 3.7 - 3.10 show the estimated k-field for each rat and each model. Finally, figure 3.11 shows another metric for comparison, the total number of tumor cells calculated at each time point.

	Rat 1		Rat 5		Rat 6		Rat 9	
$\mathcal{J}$(time point)	ISTF	FSTF	ISTF	FSTF	ISTF	FSTF	ISTF	FSTF
$\mathcal{J}(1)$	19.9	20.0	14.1	13.3	2.6	2.7	17.5	13.5
$\mathcal{J}(2)$	9.0	8.1	8.4	7.5	4.8	4.3	28.4	9.8
$\mathcal{J}(3)$	20.1	17.9	18.0	14.6	7.7	8.2	37.6	19.8
$\mathcal{J}(4)$	39.5	37.5	39.0	29.0	28.3	29.5	104.2	72.2
$\mathcal{J}(5)$	-	-	76.9	55.2	52.9	53.0	245.8	203.7
$\mathcal{J}(6)$	-	-	-	-	83.8	81.7	414.7	395.5
$\mathcal{J}(7)$	-	-	-	-	168.3	159.9	-	-

Table 3.3: Misfit results summary showing that the hyperelastic model is able to better fit the data towards later time points.

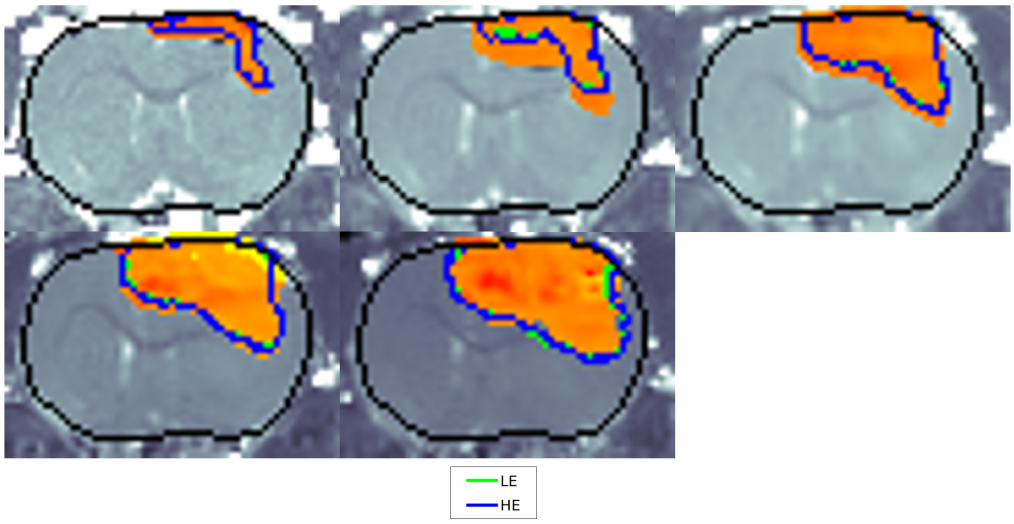

Figure 3.3: Rat 1 tumor growth accurately estimated by both models.

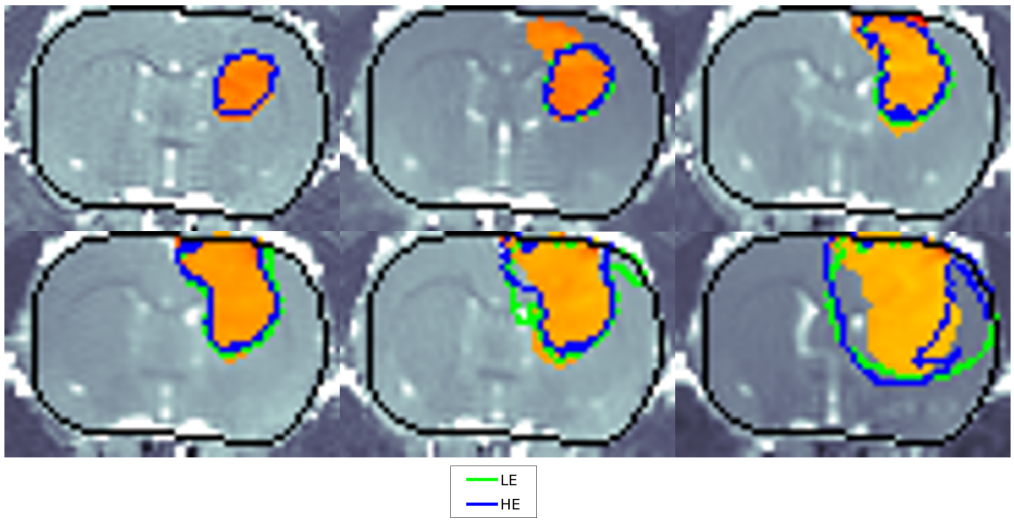

Figure 3.4: Rat 5 tumor growth estimated well by both models until the last day.

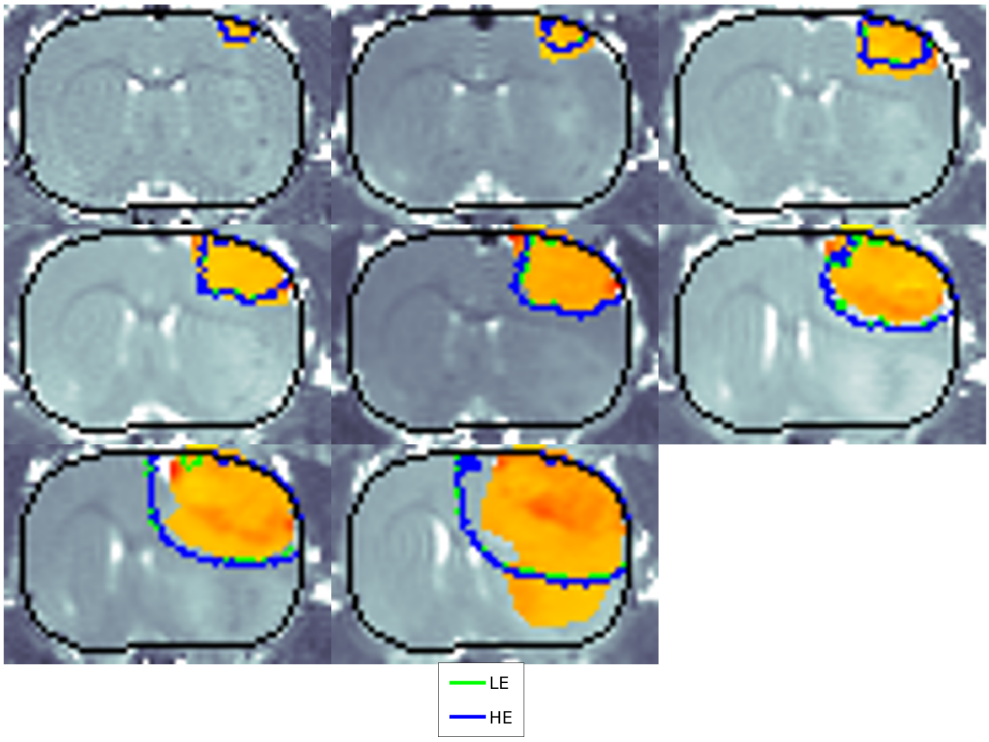

Figure 3.5: Rat 6 tumor growth estimated well by both models until the last day.

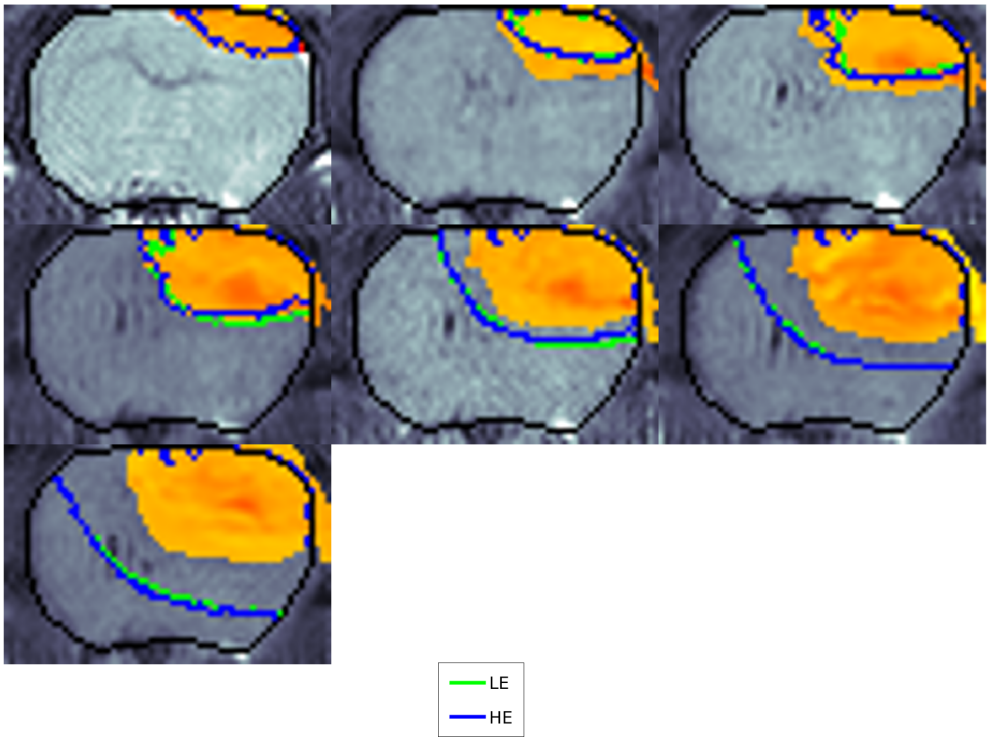

Figure 3.6: Rat 9 tumor growth modeled decently by both models until the fifth day.

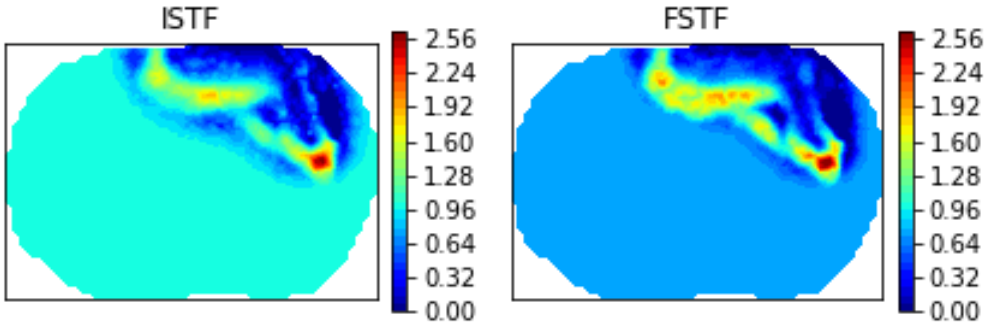

Figure 3.7: k-field estimate for rat 1

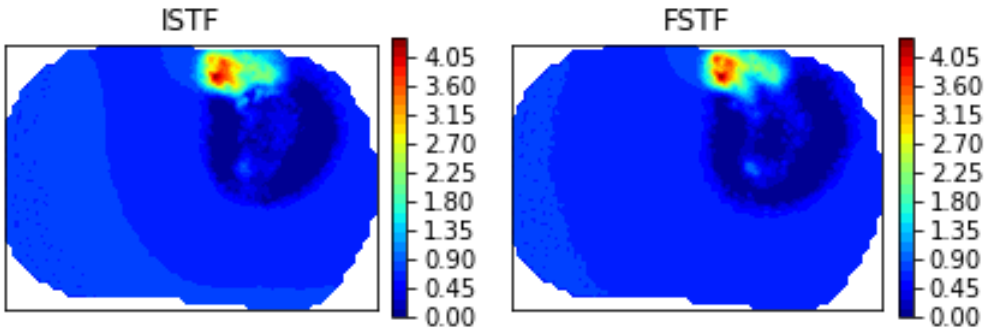

Figure 3.8: k-field estimate for rat 5

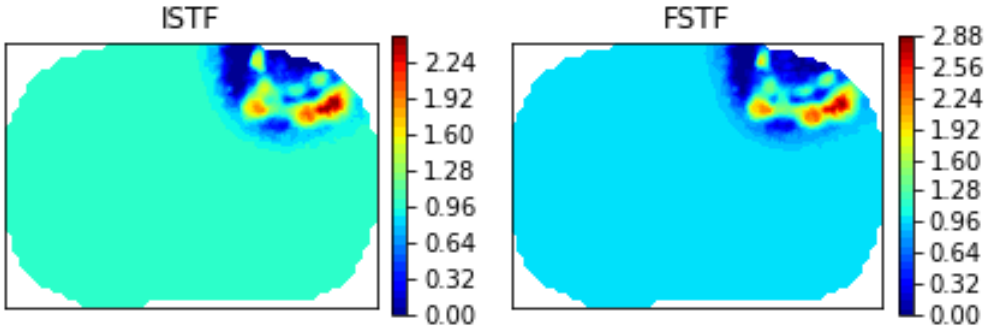

Figure 3.9: k-field estimate for rat 6

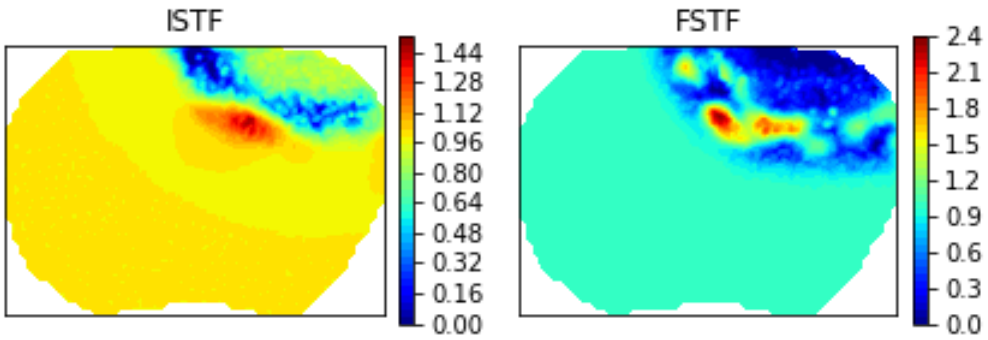

Figure 3.10: k-field estimate for rat 9

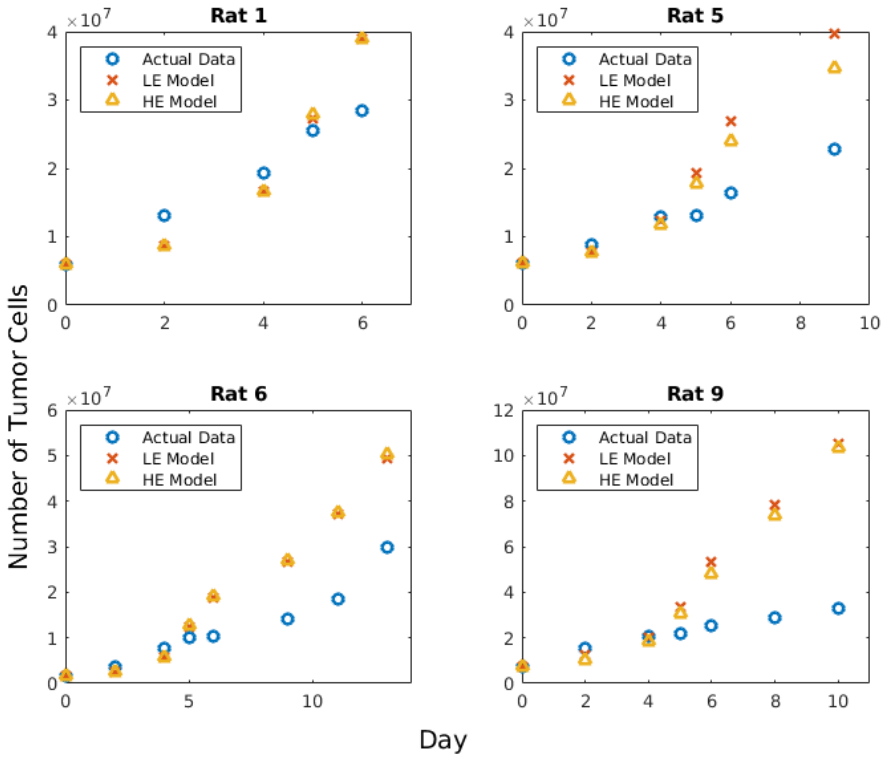

Figure 3.11: Actual versus modeled number of total tumor cells.

Another relevant result is the computational costs, which are also different even though memory used is the same. The hyperelastic model requires two non-linear solves where as the linear elastic model only requires one. This causes a doubling in time to run the parameter estimation and forward model for the FSTF model. The linear model took approximately half as long to run on average.

Chapter 4

Discussion

4.1 Summary of key results

The estimated k fields are very similar in shape and values between the two models for the first three rats, while the other estimated parameters vary more between the different models. The linear elastic model estimates more diffusion in general and predicts slightly more growth. Both models were able to accurately recover the tumor data for rat 1 with the parameter estimation procedure and less so for the other rats. The hyperelastic model had slightly better predictions of the tumor growth overall as evidenced by the data misfit and total number of cells predicted. Both models overestimated tumor cell count which is due to necrosis in the actual tumor that was not modeled here.

4.2 Study limitations

4.2.1 Models

Only two constitutive models were compared in this study, but there are many relevant nonlinear models that could be studied, including other hyperelastic models. Several of these nonlinear models do not require many parameters, some of which could be gathered from experiments. There is also the option of simplifying the nonlinear models as piecewise linear models, which would reduce the time to run the forward model.

The 2D assumption is also restrictive. The actual stresses are 3D, and the domain does not satisfy the requirements of plane strain very well. Although computation time would likely increase significantly for a 3D simulation, the results would be more realistic.

4.2.2 Parameter estimation

With regards to parameter estimation, there are a few improvements that can be made. For example, Hormuth et al. [14] tested various values of β instead of estimating it, and compared models that had the same β values. This assumption may enable better estimation of the hyperelastic model parameters.

More iterations may yield better results, as only 25 were used. The initial guesses can be randomized and several different sets used to explore the parameter space better. For the growth field k in particular, the data could be used to inform the prior so that instead of starting the parameter estimation with a uniform k-

field, it could be assumed to be zero except where the tumor seems to be growing toward. Additionally, the growth field should be estimated at different time points since as the tumor grows, it will encounter new blood vessels and grow towards them. This can be seen to happen in figure 3.5 at the last time step.

Regularization terms may improve the estimation problem. An example of where this might be relevant is in estimating the growth fields, which showed some discontinuities that would not be expected in real life. Dispersion of nutrients throughout the brain would probably be more continuous. Thus, the cost function could include the following:

$$\mathcal{J}(t) = \sum ||\phi_t(t) - \phi_{true}(t)||_\Omega^2 + C_1||\nabla k||_\Omega^2$$

where C_1 is some coefficient that would be modified for varying amounts of smoothness in the k-field.

The cost function could also be modified to give more weight to the cost of later days, since these provide more recent information on the tumor growth. Additionally, as more data is gathered, it can be used to update the parameter estimation.

Finally, the deterministic parameter estimation leads to a single prediction, while a probabilistic estimation would result in parameter distributions and uncertainty quantification.

Chapter 5

Conclusion

The work shown here indicates that for 2D tumor growth estimation using the reaction-diffusion model with mechanical effects, the current assumption of small strains is adequate. Differences between the ISTF and FSTF were slight or negligible, yet the FSTF took almost twice as long to run, rendering it less efficient. There are many small improvements to the process that can be made, however, so some further work is warranted.

Appendix A

FEniCS Implementation

```
from dolfin import * #FEA package
from dolfin_adjoint import * #Parameter estimation
import numpy as np
from scipy.io import loadmat as sc_io_loadmat
from scipy.interpolate import RegularGridInterpolator

set_log_level(ERROR) # Information display level for dolfin

class InterpolatedParameter(Expression):
    '''
        Class to get tumor cell distributions by interpolating
        based off matrices of tumor cell data
    '''
    def __init__(self,X,Y,image,**kwargs):
        self.X = X # A numpy array giving the X-spacing of the image
        self.Y = Y # Same for Y
        self.image = image # The image of measured material property
    def eval_cell(self,values,x,cell):
        interp_handle = RegularGridInterpolator((self.X,self.Y),
                                                self.image)
        values[0] = interp_handle(x)

def interp(file_loc,mat_name, norm_val):
    """
        Function to accept matlab .mat file with tumor data
        and interpolate values onto mesh
```

```
    """
    mat = sc_io_loadmat(file_loc)[mat_name]
    mat = np.fliplr(mat.T)/norm_val # Adjusted to fit the mesh
    x,y = mat.shape[0], mat.shape[1]
    mat_interp = InterpolatedParameter(np.linspace(1,x,x),
                                       np.linspace(1,y,y),
                                       mat,degree=1)
    return interpolate(mat_interp,V)

def set_nonlinear_params(param):
    # Set parameters for newton solver
    param_ns = param['newton_solver']
    param_ns['absolute_tolerance'] = 1E-7
    param_ns['relative_tolerance'] = 1E-6
    param_ns['maximum_iterations'] = 51
    param_ns['relaxation_parameter'] = 1.0
    param_ns['linear_solver'] = 'gmres'
    param_ns['preconditioner'] = 'ilu'
    param_ns['krylov_solver']['absolute_tolerance'] = 1E-8
    param_ns['krylov_solver']['relative_tolerance'] = 1E-6
    param_ns['krylov_solver']['maximum_iterations'] = 1000
    param_ns['krylov_solver']['nonzero_initial_guess'] = True

def forward(initial_p, name, record=False, annt=False, opt=False):
    """
        Here, we define the forward problem with mechanical functions

        -E(u) returns the Green-Lagrange strain tensor
        -sigma(...) returns the actual stress tensor
        -sigma_form(...) returns the stress tensor based on
          the cells (phi), elasticity coefficients, and a
          coefficient beta
        -vonmises(...) calculates the von Mises stress based
          on the actual stress tensor
    """
    I = Identity(2) # Identity tensor
    def E(u):
        return 0.5*(nabla_grad(u) + nabla_grad(u).T)
```

```
def vonmises(u):
    s = sigma(u) - (1./2)*tr(sigma(u))*I # deviatoric stress
    von_Mises = sqrt(3./2*inner(s, s))
    return project(von_Mises, V, annotate=annt)

#Set up linear elasticity problem
U = VectorFunctionSpace(mesh,'Lagrange',1)
def boundary(x, on_boundary):
    return on_boundary
bc = DirichletBC(U, Constant((0.,0.)), boundary)
p_n = interpolate(initial_p,V)
v = TestFunction(U)

ffc_options = {"quadrature_degree": 2, 'cpp_optimize': True}

if lin_hyp == 0: # Linear model
    def sigma(u):
        s = 2*mu*E(u)+lmbda*tr(E(u))*I
        return s
    u = TrialFunction(U)
    a = inner(2*mu*E(u)+lmbda*tr(E(u))*I,E(v))*dx
    L = inner(2*beta*p_n*I*(mu+lmbda),E(v))*dx
    u = Function(U, annotate=annt)
    def mech():
        solve(a == L, u, bc,
                 form_compiler_parameters=ffc_options,
                 annotate=annt)
        return u
else: # Nonlinear model
    def sigma(u):
        F = I + grad(u) # Deformation gradient
        B = F*F.T
        C = F.T*F
        J = det(F)
        I1 = tr(C)
        s = lmbda*(J-1)*I+mu*(B-1./2*I1*I)/(J**(5./3))
        return s
    def sigma_form(u, phi):
```

```
        F = I + grad(u)
        Fs = F/(1+beta*phi)
        Bs = Fs*Fs.T
        Js = det(Fs)
        return 1/(1+beta*phi)*(mu/(Js**(5./3))*(Bs-1./2*tr(Bs)*I)
                              +lmbda*(Js-1)*I)

    u = Function(U, annotate=annt)
    du = TrialFunction(U)
    F_HE = inner(sigma_form(u, p_n), E(v))*dx
    J_HE = derivative(F_HE,u,du)
    problem_HE = NonlinearVariationalProblem(F_HE, u, bc,
                            J=J_HE,
                            form_compiler_parameters=ffc_options)
    solver_HE = NonlinearVariationalSolver(problem_HE)
    param_HE = solver_HE.parameters
    set_nonlinear_params(param_HE)
    def mech():
        solver_HE.solve(annotate=annt)
        return u

# First iteration solving for displacement,
# and using the von mises stress field for D
disp = mech()
vm = vonmises(disp)
D = project(D0*exp(-gammaD*vm),V,annotate=annt)

# Set up reaction-diffusion problem
dp = TrialFunction(V)
p = Function(V,annotate=annt)
q = TestFunction(V)
F_RD = (1/dt)*(p - p_n)*q*dx + D*dot(grad(q),grad(p))*dx
      - k*p*(1 - p)*q*dx
J_RD = derivative(F_RD,p,dp)

for n in range(num_steps1): # Run model up to first time point
    # Solve reaction diffusion
    problem_RD = NonlinearVariationalProblem(F_RD, p,
```

```
                            J=J_RD,
                            form_compiler_parameters=ffc_options)
    solver_RD = NonlinearVariationalSolver(problem_RD)
    param_RD = solver_RD.parameters
    set_nonlinear_params(param_RD)
    solver_RD.solve(annotate=annt)
    p_n.assign(p)

    # Solve for displacement and vonmises stress
    disp = mech()
    vm = vonmises(disp)
    D = project(D0*exp(-gammaD*vm),V,annotate=annt)
est1 = Function(V, annotate=False)
est2 = Function(V,annotate=False)
est1.vector()[v2d] = p.compute_vertex_values()

if opt: # Run model to second time point when optimizing
    for n in range(num_steps2):
        # Solve reaction diffusion
        problem_RD = NonlinearVariationalProblem(F_RD, p,
                            J=J_RD,
                            form_compiler_parameters=ffc_options)
        solver_RD = NonlinearVariationalSolver(problem_RD)
        param_RD = solver_RD.parameters
        set_nonlinear_params(param_RD)
        solver_RD.solve(annotate=annt)
        p_n.assign(p)

        # Solve for displacement and vonmises stress
        disp = mech()
        vm = vonmises(disp)
        D = project(D0*exp(-gammaD*vm),V,annotate=annt)
    est2.vector()[v2d] = p.compute_vertex_values()

    return est1, est2

else:
    return est1
```

```
# Callback function for the optimizer
# Writes intermediate results to a logfile
def eval_cb(j, m):
    """ The callback function keeping a log """
    f_log.write("objective = %15.10e \n" % j)

def objective(est1, est2, targ1, targ2):
    return assemble(inner(est1-targ1, est1-targ1)*dx)
          + assemble(inner(est2-targ2, est2-targ2)*dx)

def optimize(dbg=False):
    # Define the control
    m = [Control(D0), Control(k), Control(gammaD), Control(beta)]

    # Execute first time to annotate the tape
    p1, p2 = forward(initial_p, 'annt', False, True, True)

    Obj = objective(p1, p2, second_p, third_p)

    # Prepare the reduced functional
    rf = ReducedFunctional(Obj,m,eval_cb_post=eval_cb)

    # upper and lower bound for the parameter field
    D_lb = 0.
    D_ub = 5.
    k_lb, k_ub = Function(V,annotate=False), Function(V,annotate=False
       )
    k_lb.vector()[:] = 0.
    k_ub.vector()[:] = 5.
    gD_lb = 0.01
    gD_ub = 5
    b_lb = 0.01
    b_ub = 10.
    bnds = [[D_lb, k_lb, gD_lb, b_lb],[D_ub, k_ub,gD_ub,b_ub]]

    # Run the optimization
    m_opt = minimize(rf,method='L-BFGS-B', bounds=bnds,
```

```
                    options={"disp":True,
                             "gtol":2.0e-5,
                             "ftol":2.0e-7,
                             "maxiter":25,
                             "maxls": 15})

    return m_opt

######################################################################
# MAIN
######################################################################

# Whether to run linear or nonlinear
lin_hyp = 0

# Prepare a mesh
mesh = Mesh("gmsh.xml")
V = FunctionSpace(mesh, 'CG', 1)
v2d = vertex_to_dof_map(V)

# Days data and time steps
days = [0,2,4,5,6] # Days rats were examined at

# Constant inputs for optimization
theta = 50970. # carrying capacity - normalize cell data
mu = .42 # kPa, bulk shear modulus
nu = .45
lmbda = 2*mu*nu/(1-2*nu)

init_day = 0 # Day to use for initial data
second_day = 1
third_day = 2

# Write vector results in one file:
# - forward: values written at each day values (not in between)
f_forward = XDMFFile("forward.xdmf")
f_forward.parameters["flush_output"] = True
f_forward.parameters["functions_share_mesh"] = True
```

```
# Model parameters: initial guesses
gammaD = .5
beta = .5
D0 = .5
k = project(Constant(1.),V,annotate=False)

t0 = days[init_day] # Current day
t1 = days[second_day] # Current day
t2 = days[third_day] # Next day
T1 = t1-t0 # Final time is dt between days
T2 = t2-t1 # Final time is dt between days
num_steps1 = T1*10 # number of time steps
num_steps2 = T2*10 # number of time steps
dt = T1/float(num_steps1) # time step size

# Load tumor data: from current day and next day
initial_p = interp("tumor_t"+str(t0)+".mat","tumor",theta)
second_p = interp("tumor_t"+str(t1)+".mat","tumor",theta)
third_p = interp("tumor_t"+str(t2)+".mat","tumor",theta)

# Optimization
[D0, k, gammaD, beta] = optimize() # optimize these params using the
                                   # adjoint method

# Compare optimized tumor growth to actual at several time points.
model_p = initial_p # Initialize
model_p.rename('opt_p','optimized tumor')
f_forward.write(model_p, float(t0))

true_p = initial_p # Initialize
true_p.rename('true_p', 'actual tumor')
f_forward.write(true_p, float(t0))

newdays = days[init_day:]
for idx,day in enumerate(newdays[:-1]):
    step = newdays[idx+1]-newdays[idx]
    num_steps = step*10 # number of time steps
```

```
    dt = step/float(num_steps) # time step size

    # Run forward model using optimized values
    model_p = forward(model_p,"Null",False,False)
    model_p.rename('opt_p','optimized tumor')
    f_forward.write(model_p,float(newdays[idx+1]))

    # Save actual tumor for comparison
    true_p = interp("tumor_t"+str(newdays[idx+1])+".mat","tumor",theta
        )
    true_p.rename('true_p','actual tumor')
    f_forward.write(true_p,float(newdays[idx+1]))

    # Save J_opt
    print('J_opt start '+str(t0)+' day '+str(newdays[idx+1])+' = '
              +str(objective(model_p,true_p,true_p,true_p))+'\n')
```

Appendix B

On Bayesian Methods for Model Calibration and Selection

B.1 Model Calibration

When attempting to describe phenomena with mathematical models, models are often chosen based on certain expected behavior from the phenomena in question. When there are no such expectations due to lack of knowledge or understanding of the phenomena, or to supplement hypotheses about the models, experiments are conducted, the data is examined, and assumptions are made or refined based on the results. At this point, an effort is made to determine more specific characteristics of the model that allow it to best fit the data. This is called model calibration. Regression analysis is used in this case to determine the relationships between the different variables. There are different forms of regression analysis, and here we will describe two standard methods: ordinary least squares and Bayesian inference. The latter inherently allows for uncertainty in the parameters and model and can thus provide more information when making predictions.

B.1.1 Ordinary Least Squares (OLS)

Given a mathematical model and data, OLS is a method of finding coefficients which minimize sum of squared differences between the model and the data points. Given data points $\boldsymbol{d}(\boldsymbol{x})$ evaluated at the independent variables $\boldsymbol{x}$ and a mathematical model $y(\boldsymbol{\theta})$, the least squares approach is to find model parameters $\boldsymbol{\theta}$ which most closely align the model with the data:

$$\boldsymbol{\theta} = \text{argmin}(\boldsymbol{y}(\boldsymbol{\theta}, \boldsymbol{x}) - \boldsymbol{d}(\boldsymbol{x}))^2 \tag{B.1}$$

This is an oft-used and therefore deeply studied method with various ways of solving that will not be discussed here. A key point is that this is a frequentist approach which relies solely on the data, and so there must be sufficient data. There are several measures of "goodness-of-fit" of the model, but these are flawed in that they may recommend a more complex model than the true model in order to fit the noisy data. For example, as shown in figure B.1, noise can scatter linear data and be best fit by a high-order polynomial.

B.1.2 Bayesian Inference

Bayesian inference improves on OLS by making use of prior information in addition to the data. Furthermore, instead of producing a set of optimal values, it produces probability distributions $\pi_{post}(\boldsymbol{\theta})$ of the parameters. That is, Bayesian inference provides information on the uncertainty in each parameter given uncertainty in the data and model.

This method has been described as “reallocation of credibility among possibilities" [19]. Given some set of possibilities described by a prior distribution $\pi_{prior}(\boldsymbol{\theta})$, a likelihood functions $\pi_{like}(\boldsymbol{d}|\boldsymbol{\theta})$ serves to shift the probabilities to better align the data, producing a posterior distribution $\pi_{post}(\boldsymbol{\theta})$. An example of this reallocation is shown in figure B.2, where the data serves to shift the posterior from a mode near .2 to a mode near .5. The mathematical description of the posterior is calculated using Bayes' theorem,

$$\pi_{post}(\boldsymbol{\theta}|\boldsymbol{d}) = \frac{\pi_{like}(\boldsymbol{d}|\boldsymbol{\theta})\pi_{prior}(\boldsymbol{\theta})}{\pi(\boldsymbol{d})} \tag{B.2}$$

Selecting a prior

The first steps of Bayesian inference include identifying the data, the models, and the prior distributions. The prior can be dependent or independent of the data. For example, for model comparison, which will be described later, it is recommended that a portion of the data be used to determine prior distributions for all the models. Alternatively, the priors can be developed from some subjective beliefs or from previous data. A uniform prior can also be used as a "vague", or noninformative, prior, which effectively gives all the possible parameter values equal probability. An example of an informative prior would be:

$$\pi_{prior}(\theta) \sim \mathcal{N}(0, 1)$$

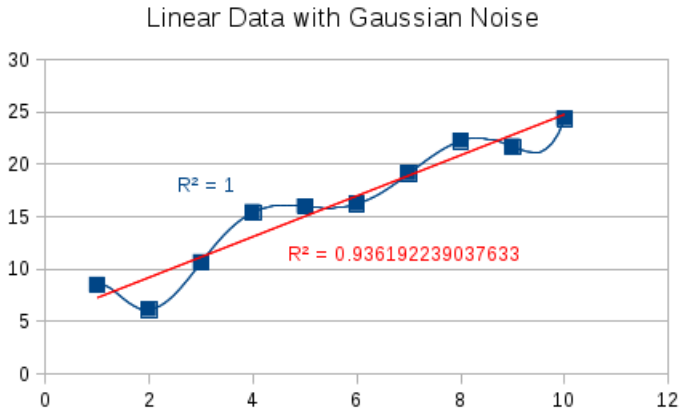

Figure B.1: Deterministic inference using linear and polynomial models.

Likelihood function

The likelihood function stems from a statistical description of the model. It describes the likelihood of observing the data given a set of parameter values. At time it may arise from simple observation of the process, while at other times assumptions may have to be made. For example, the likelihood of a specific set of coin flips given the coin's bias can be described by a Bernoulli distribution. In other cases it might be assumed that the data follow a Gaussian distribution and so a corresponding likelihood function would be used. If the data is independent and identically distributed, the likelihood would be as follows:

$$\pi_{like}(\boldsymbol{d}|\theta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(d_i - \theta)^2\right]$$
$$= \frac{1}{\left(\sqrt{2\pi\sigma^2}\right)^n} \exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(d_i - \theta)^2\right] \tag{B.3}$$

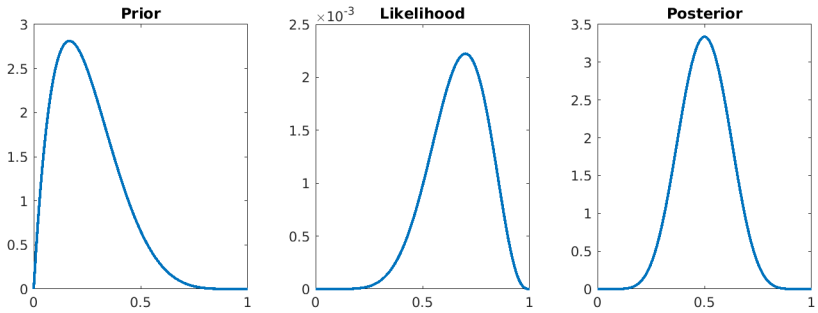

Figure B.2: An example of reallocating credibility from a prior distribution, using a likelihood function that represents the data, to a posterior distribution.

B.1.3 Algorithms for sampling a posterior distribution

With a prior and likelihood, the posterior can be determined. The denominator in the right hand side can be difficult to calculate but is not necessary. The reason is that since the posterior is a distribution, the integral of a distribution over the domain is unity, and the denominator is independent of θ, the denominator is simply a normalizing constant that can be deduced later or ignored if the end goal is determining the shape of the distribution. While there may not be a closed-form solution for calculating the numerator analytically, algorithms exist for numeri-

cal solutions. QUESO (Quantification of Uncertainty for Estimation, Simulation, and Optimization) is a collection of algorithms and C++ classes, including some that can be applied to Bayesian inference problems, that has been used in this research [6]. In particular, QUESO uses the Delayed Rejection Adaptive Metropolis (DRAM) algorithm when the form of the posterior distribution is desired and not necessarily the normalized distribution [11]. For model selection, the Adaptive Multilevel Stochastic Simulation Algorithm (AMSSA) approximates the value of the denominator [26].

Metropolis Algorithm

The DRAM algorithm builds upon the classic Metropolis-Hastings (MH) algorithm that was developed in 1953. The classic MH algorithm begins at a random or prescribed point inside the parameter space with a corresponding "probability" determined by the numerator (although it is not actually a probability since the numerator describes a non-normalized distribution, it can be thought of in the same vein.) A second sampling point is generated randomly and it's probability computed. If it has a higher probability than the original point, the algorithm moves to that point. If the probability is lower, the algorithm moves conditionally, with a probability of moving equal to the ratio of probabilities. See listing B.1 for a sample implementation in MATLAB.

It can be shown that following this procedure indefinitely will produce the actual posterior distribution; that is, the number of times a point is sampled with respect to the total number of samples is its actual posterior probability. Thousands

of iterations usually suffice for an approximation of the posterior. See the following example.

Metropolis Hastings Example: Using the prior and likelihood of figure B.2, a few steps of this procedure are detailed below and shown in figure B.3:

1. Suppose a random starting point of $\theta_1 = .3$.
2. Evaluate $\pi_{like}(d|\theta = .3)\pi_{prior}(\theta = .3) = .7068$
3. Use a random number generator to pick the next θ_2. Suppose $\theta_2 = .5$.
4. Evaluate $\pi_{like}(d|\theta = .49)\pi_{prior}(\theta = .49) = 3.3278$. Since $\theta_2 > \theta_1$, "move" to this value and record it in the chain.
5. Use a random number generator to pick the next θ_2. Suppose $\theta_3 = .8$.
6. Evaluate $\pi_{like}(d|\theta = .8)\pi_{prior}(\theta = .8) = .094$. Since $\theta_3 < \theta_2$, the probability of "moving" to this value is $\frac{.094}{3.3278} = .0282$. A random number generator can be used to decide whether to move or stay and pick another θ for evaluation.

This algorithm is an example of a Markov Chain Monte Carlo algorithm, named as such because it produces a "chain" of samples (also known as a Markov Chain), and because Monte Carlo methods include those for random sampling. Figure B.4 shows both a sample chain and the resulting histogram, a discrete estimate of the posterior. Several chains can be produced and used to estimate the

posterior.

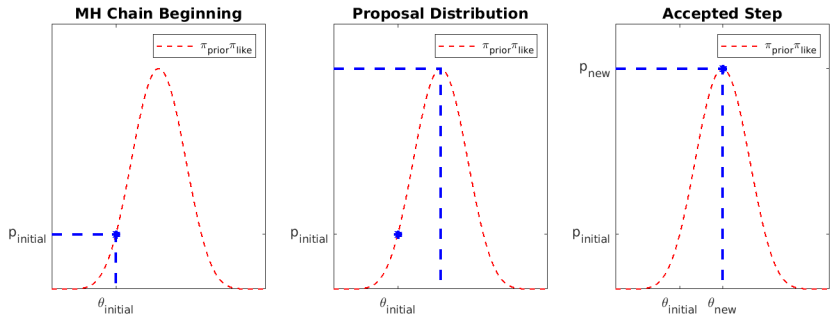

Figure B.3: Steps 1-4 of the the procedure described above for the distributions in figure B.2.

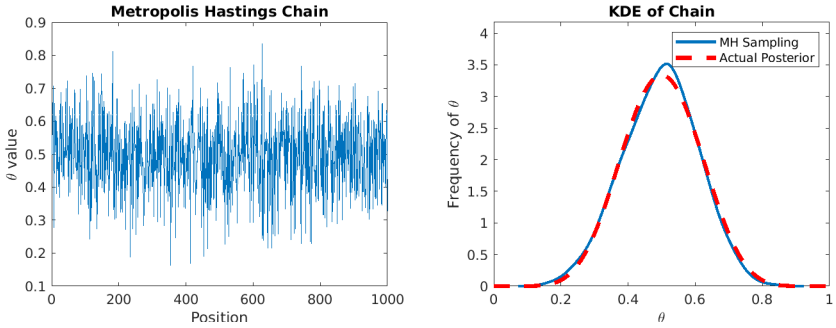

Figure B.4: An example of the resulting chain and posterior distribution.

Listing B.1: Metropolis Hastings Algorithm

```
% Example of MH with a known beta posterior distribution
chain(1) = rand; % Begin chain within (0,1)
for i = 1:1000
    sample = betapdf(chain(i),a,b); % Current sample
    next = rand;
    test = betapdf(next,a,b); % Sample at test point
```

```
    p = test/samp;  % Calculate ratio
    if rand < p   % Determine whether to move
        chain(i+1) = next; % Conditional move
    else
        chain(i+1) = chain(i); % Conditional stay
    end
end
```

The first improvement on this algorithm is improving the method in which steps are proposed. Instead of using a uniform distribution, i.e. taking a random value from the parameter space, the next step can be chosen from a Gaussian distribution about the current step. Proposed values are likely to be closer to the current value, making the process a random walk. See figure B.5 for an example of what a proposal distribution would look like.

Using a proposal distribution with multiple shape parameters introduces the question of what shape the distribution should take. The selection could have a significant effect on the resulting chain. For example, when inferring for a single parameters, a Gaussian distribution with a very small standard deviation will restrict movement and potentially prevent the entire parameter space from being sampled properly. A very large standard deviation results in an essentially uniform distribution that is no longer a random walk. When inferring for multiple parameters, orientation of the proposal distribution must be considered as well.

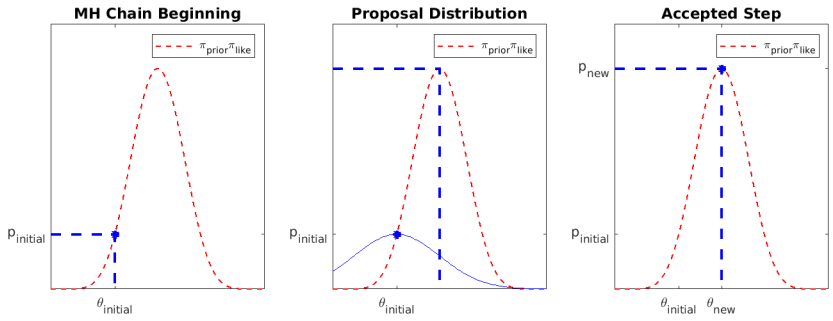

Figure B.5: Using a Gaussian proposal distribution to propose subsequent steps.

One answer to this question is to adapt the distribution as the chain is developed, increasing the standard deviation if the walk is exploring the parameter space too slowly or vice-versa. This is the Adaptive Metropolis method [10]. After setting an initial proposal covariance, the standard MH algorithm is run, and every t iterations the proposal covariance is updated,

$$cov_t = s_d(cov(x_1, ..., x_{t-1}) + \epsilon I)$$

The new proposal covariance is the covariance of the chain scaled by some factor based on the dimension of the problem, $s_d = 2.4^2/d$. The ϵ variable is some small number meant to prevent the covariance matrix from becoming singular. The effect of these adaptations is to decrease the covariance if too many steps are being rejected, thus increasing the chance of acceptance, and otherwise raising the covariance to some scaled value of the actual posterior's covariance, which has been reported as optimal by Gelman et al. [9]. See listing B.2 for possible implementation.

Listing B.2: Adaptive Metropolis Algorithm

```
% Example of AM with a known beta posterior distribution
chain(1) = rand; % Begin chain within (0,1)
s = .1;  % Initial standard deviation
adaptint = 20; % When to adapt
adaptscale = 2.4; % Scale for 1D
for i = 1:1000
    if mod(i,adaptint) == 0
        s = (std(chain(1:i))+1e-5)*adaptscale; end
    samp = betapdf(chain(i),a,b); % Current sample
    next = chain(i)+s*randn;   % Gaussian proposal
    test = betapdf(next,a,b);
    p = test/samp;
    if rand < p
        chain(i+1) = next;
    else
        chain(i+1) = chain(i);
    end
end
```

The next improvement comes from reducing the amount of rejections. This reduces autocorrelation and allows for greater exploration of the space. The method is the Delayed Rejection algorithm [25]. When a proposed step is rejected, instead of restarting from the current value, the next step is proposed from the rejected

step and compared to the current value. This can be repeated multiple times, or "stages". The probability of acceptance at these stages is modified to keep the Markovian property of the chain. See listing B.3 for an example of the implementation.

Listing B.3: Delayed Rejection Algorithm

```
% Example of Delayed Rejection with a known beta posterior distribution
chain(1) = rand; % Begin chain within (0,1)
s = .1;  % Initial standard deviation
drscale = .5; % Standard deviation scale for secondary proposal
for i = 1:1000
    samp = betapdf(chain(i),a,b); % Current sample
    next = chain(i)+s*randn;   % Gaussian proposal
    test = betapdf(next,a,b);
    p = test/samp;
    if rand < p
        chain(i+1) = next;
    else
        next2 = n+drscale*s*randn; % Second Gaussian proposal
        test2 = betapdf(next2,a,b);
        q1zy = normpdf(next,next2,s);
        q1xy = normpdf(next,chain(i),s);
        alpha2 = min(1,test/test2);
        p2 = (test2/samp)*(q1zy/q1xy)*((1-alpha2)/(1-p));
```

```
        if rand < p2
            chain(i+1) = next2;
        else
            chain(i+1) = chain(i);
        end
    end
end
```

Finally, using Delayed Rejection with Adaptive Metropolis results in an algorithm that has fewer rejections and better exploration of the sample space, leading to a larger effective sample size.

Adaptive Multilevel

The algorithm used by QUESO for calculation of model evidence, which is useful for model selection (see section B.2), uses DRAM but in a different way than immediately solving for the posterior. The idea is to incrementally solve for the posterior beginning with the prior and slowly adding the data through the likelihood function. Instead of sampling from $\pi_{like}(\boldsymbol{d}|\boldsymbol{\theta})\pi_{prior}(\boldsymbol{\theta})$, it begins with the prior and slowly adds the likelihood,

$$\pi^n_{post}(\boldsymbol{\theta}|\boldsymbol{d}) = \pi_{like}(\boldsymbol{d}|\boldsymbol{\theta})^{\tau_n}\pi_{prior}(\boldsymbol{\theta})$$

where $0 = \tau_1 < \tau_2 < ... < \tau_N = 1$. At each new level, the previous posterior is used to generate starting points for the chains. The "model evidence", $\pi(\boldsymbol{d})$, can

then be determined in the following manner,

$$\begin{aligned}
\pi(\boldsymbol{d}) &= \int \pi_{like}(\boldsymbol{d}|\boldsymbol{\theta})\pi_{prior}(\boldsymbol{\theta})\, d\boldsymbol{\theta} \\
&= \int \pi_{like}(\boldsymbol{d}|\boldsymbol{\theta})^{(1-\tau_{N-1})} ... \pi_{like}(\boldsymbol{d}|\boldsymbol{\theta})^{\tau_0}\pi_{prior}(\boldsymbol{\theta})\, d\boldsymbol{\theta} \\
&= c_1 \int \pi_{like}(\boldsymbol{d}|\boldsymbol{\theta})^{(1-\tau_{N-1})} ... \frac{\pi_{like}(\boldsymbol{d}|\boldsymbol{\theta})^{\tau_0}\pi_{prior}(\boldsymbol{\theta})}{c_1}\, d\boldsymbol{\theta} \\
&= c_N ... c_1
\end{aligned}$$

Where e.g. $c_1 \equiv \int \pi_{like}(\boldsymbol{d}|\boldsymbol{\theta})^{\tau_0}\pi_{prior}(\boldsymbol{\theta})\, d\boldsymbol{\theta}$ can be solved using Monte Carlo with small τ_n since the prior integrates to 1. Since the values c_n can be extremely small or large, it is customary to work with the log of the evidence,

$$\ln(c_N ... c_1) = \ln(c_N) + ... + \ln(c_1)$$

Figure B.6 shows an example of intermediate posteriors used to calculate the evidence, as well as the final distribution and evidence.

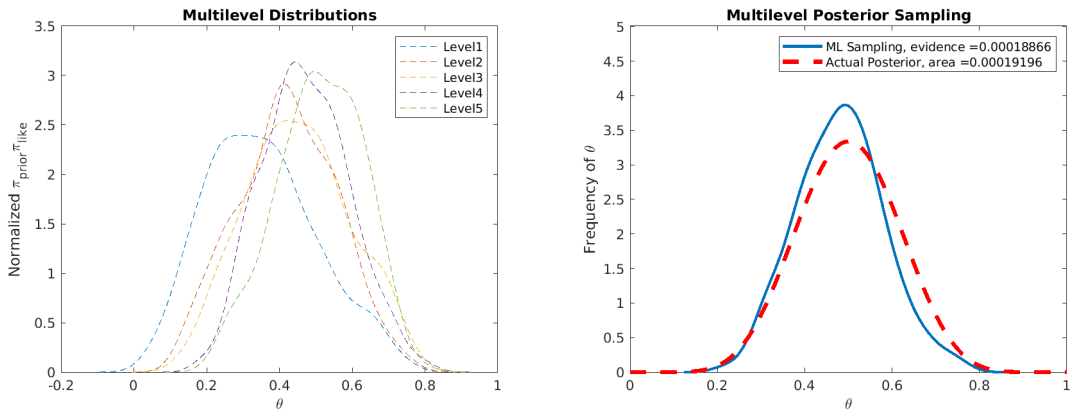

Figure B.6: Left: An example of intermediate levels using AMSSA for a known beta function. Right: The final level and calculated evidence compared to actual evidence.

B.2 Model selection

Bayesian inference can also be used to compare models and determine which model is more likely given the data. For the comparison, not only is the model fit taken into account, but also the complexity of the model. For example, in the case of noisy linear data discussed in section B.1.1, the linear model might have a greater probability despite not being the best fit because it's less complex.

B.2.1 Hierarchical models

To understand model selection it is important to first describe hierarchical models, models where there are multiple parameters that may depend on each other in some sort of hierarchy. A relevant example would be determining the

growth rate of tumors. Although one could devise a model using all of the data, it might be more realistic to assume that different tumor types may have different growth rates. Then, one would be inferring for the growth rates of each tumor (multiple parameters) as well as the growth rate of all tumors (a single parameter). Algorithms for this form of Bayesian inference are also called "multilevel" algorithms, such as the Adaptive Multilevel algorithm described above.

B.2.2 Model comparison

One can regard a level of different models as one set of parameters, each with their own subset of parameters. Similar to how Bayesian inference for a single model provides posterior probabilities for different parameters values, each model will have some posterior probability assigned to it. Comparison of these probabilities can help with model selection.

The calculations for these values are as follows. Suppose there are models m_j, $j = 1, ..., n$. Each model may have a set of parameters $\boldsymbol{\theta}_j$. Given the parameter likelihood functions, $\pi_{like,j}(\boldsymbol{d}|\boldsymbol{\theta}_j, m_j)$, the parameter priors $\pi_{prior,j}(\boldsymbol{\theta}_j|m_j)$, and the model priors $\pi_{prior}(\boldsymbol{m})$, and using the hierarchy of parameter dependence, the posterior is:

$$\pi_{post}(\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_n, m_j|\boldsymbol{d}) = \frac{\pi_{like,j}(\boldsymbol{d}|\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_n, m_j)\pi_{prior,j}(\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_n, m_j)}{\pi(\boldsymbol{d})} \tag{B.4}$$

The denominator can be rewritten using the theorem of total probability,

$$\begin{aligned}\pi(\boldsymbol{d}) &= \sum_j \pi_{like,j}(\boldsymbol{d}|\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_n, m_j)\pi_{prior,j}(\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_n, m_j) \\ &= \sum_j \int \pi_{like,j}(\boldsymbol{d}|\boldsymbol{\theta}_j)\pi_{prior,j}(\boldsymbol{\theta}_j, |m_j)\pi_{prior}(m_j)\, d\boldsymbol{\theta}_j\end{aligned}$$

Examining a single model gives:

$$\pi(m_j|\boldsymbol{d}) = \frac{\pi(\boldsymbol{d}|m_j)\pi_{prior,j}(m_j)}{\pi(\boldsymbol{d})}$$

Where the model evidence $\pi(\boldsymbol{d}|m_j) = \int \pi(\boldsymbol{d}|\boldsymbol{\theta}_j, m_j)\pi(\boldsymbol{\theta}_j|m_j)\, d\,\boldsymbol{\theta}_j$ is the normalizing constant that is in the denominator when considering a single model (see equation B.2). It is thus possible to compare models by calculating the ratios of their posterior probabilities since, if we assign each model equal prior probability, the ratio between posterior probabilities of each model is simply:

$$\frac{\pi(m = 1|\boldsymbol{d})}{\pi(m = 2|\boldsymbol{d})} = \frac{\pi(\boldsymbol{d}|m = 1)}{\pi(\boldsymbol{d}|m = 2)} \tag{B.5}$$

When determining the probability of each model compared to the entire set of models, the calculation involves comparison to the total probability,

$$\text{model possibility} = \frac{\pi(\boldsymbol{d}|m = 1)}{\sum_n \pi(\boldsymbol{d}|m = n)} \tag{B.6}$$

While computing model possibility may not be possible if the model evidence is extremely small or large when calculated using QUESO, model compari-

son can still be used.

B.2.3 Model Complexity

As a final remark for this section, it can be seen from the prior calculations that model complexity is inherently penalized using Bayesian inference. As discussed in section B.1.1, a more complex model may result in a better fit with the data despite not being the true underlying model. However, in Bayesian inference, such a model will have priors for each parameters. This serves to "dilute" the prior or spread it out and weaken it, such that although the data will fit the model well and cause a sharp likelihood, the shallow prior will balance this out. The resulting posterior may not be as sharp as that of a less complex model. The example in the following section shows this concept in practice.

B.3 Examples of Calibration and Selection

QUESO includes several examples for applying these algorithms. These were studied to gain familiarity with the problem setup and C++ classes, and then the algorithms were used on an example similar to what is expected for this research. The anticipated application is determining when a linear elastic (LE) assumption of material behavior is appropriate, and whether there is a point when nonlinear behavior is present and significant enough to warrant a different model.

To study how Bayesian inference could be applied to such a problem, data was acquired of stainless steel specimens under axial tension, which exhibit linear behavior initially and then varying amounts of nonlinear behavior. Figure B.7

shows the expected strain as a function of stress along with the strain's standard deviation. To briefly expand on the relation between this example and the current research: many materials exhibit similar behavior, with small stresses being linearly proportional to strain and elastic (meaning that when the stress is reduced, the strain returns to zero), and larger stresses being nonlinearly proportional and plastic (meaning that some permanent strain remains). Brain and tumor tissues in particular are hypothesized to behave similarly, and with the significant deformations seen *in vivo* it is further hypothesized that a nonlinear model will improve tumor growth predictions.

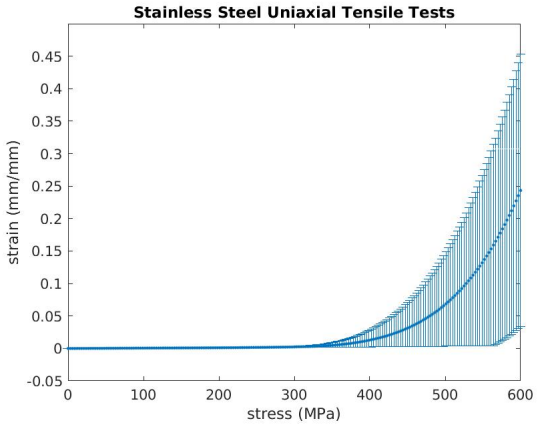

Figure B.7: Mean and variance of stainless steel specimens under axial tension.

The LE model is:

$$\epsilon = \frac{\sigma}{E}$$

where ϵ is strain, σ is stress, and E is some constant which is often called the modulus of elasticity. The nonlinear model used is the Ramberg-Osgood (RO)

model:

$$\epsilon = \frac{\sigma}{E} + Y\left(\frac{\sigma}{\sigma_y}\right)^n$$

where the new constant σ_y is the 'yield stress' and the other constants determine the curvature.

B.3.1 OLS Solutions

As a starting point, OLS gives the solutions listed in table B.1 for these models and data. These solutions produce the curves shown in figure B.8.

	E	Y	σ_y	n
LE	186,730	-	-	-
RO	186,806	.004	373	9.03

Table B.1: Ordinary Least Squares solutions for data from figure B.7.

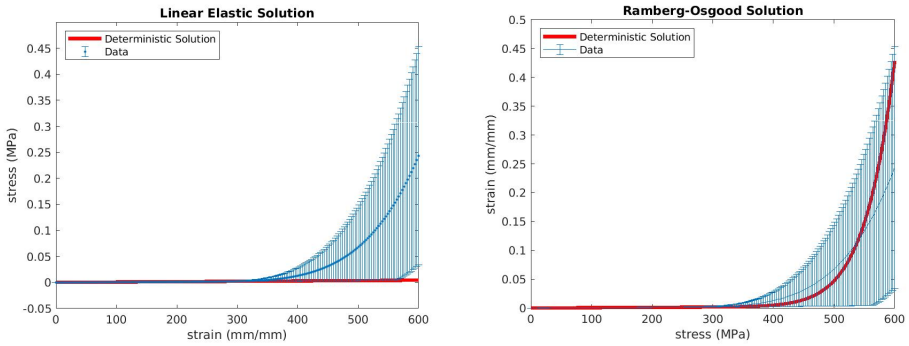

Figure B.8: Left: OLS solution to LE model used for forward statistical problem. Right: OLS solution to RO model used for forward statistical problem.

B.3.2 Bayesian Inference Solutions

The Bayesian inference problem begins with an assumption that the data is normally distributed about the mean. This means the likelihood will be in the same form as equation B.3 For the priors, it is possible to use subjective beliefs/judgement, or a non-informative prior can be used in conjunction with some portion of the data (typically $\sim$10%) to produce a posterior that can then be used as a prior for the remainder of the data. The latter procedure results in priors that are more fair for model selection/comparison purposes. In this case, some priors that might result are:

$$\begin{aligned}
\text{LE Prior: } \pi_{prior}(E) &\sim \mathcal{U}(1000, 300000) \\
\text{RO Prior: } \pi_{prior}(E) &\sim \mathcal{U}(160000, 240000) \\
\pi_{prior}(Y) &\sim \mathcal{U}(.0001, .01) \\
\pi_{prior}(\sigma_y) &\sim \mathcal{U}(200, 600) \\
\pi_{prior}(n) &\sim \mathcal{U}(4, 16)
\end{aligned}$$

It can be seen that the LE prior for E spans a larger range due to the data being nonlinear. With these inputs, QUESO will run the AMSSA algorithm and output a chain of posterior samples. The results are given in figures B.9 and B.10. It can be seen that the posteriors are very sharp, showing very little variance. This results in forward solutions that look similar to the OLS solutions: near to the average and showing very little variance (in this case undetectable to the human eye).

It is believed that the reason for such poor results (poor in that they don't reflect the actual variance at high stress levels) lies in the choice of likelihood, presented again here:

$$\pi_{like}(\boldsymbol{d}|\theta) = \frac{1}{\left(\sqrt{2\pi\sigma^2}\right)^n} \exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(d_i-\theta)^2\right]$$

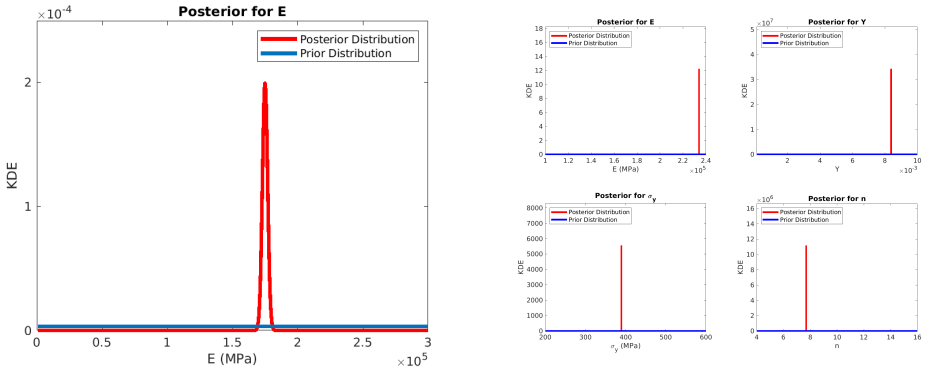

Figure B.9: Left: Bayesian inference solution to LE model. Right: Bayesian solution to RO model.

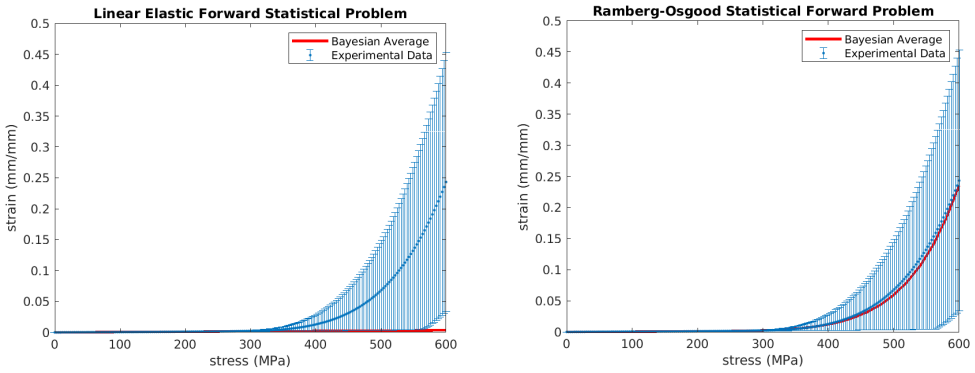

Figure B.10: Left: Bayesian inference solution to LE model used for forward statistical problem. Right: Bayesian inference solution to RO model used for forward statistical problem.

The likelihood involves dividing by the standard deviation squared, which is extremely small for much of the initial linear data. This affects the posterior by giving much more weight to the linear data and resulting in a very tight range of possible parameter values. One alternative that has been suggested is inferring for the standard deviation as a separate parameter, of which there are examples of in the QUESO manual. A single standard deviation value would show similar amounts of variance at all stress levels, though, so multiple standard deviations were inferred for here. For example, in the stress range 0-200, a single standard deviation might describe the data well; another in the 200-350 range, another in the 350-400 range, and so on. The results of this assumption are shown in figures B.11 and B.12.

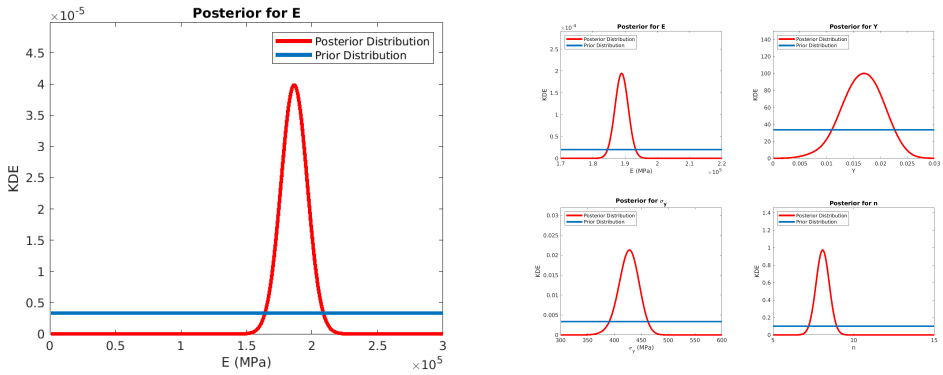

Figure B.11: Left: Updated Bayesian inference solution to LE model. Right: Updated Bayesian solution to RO model.

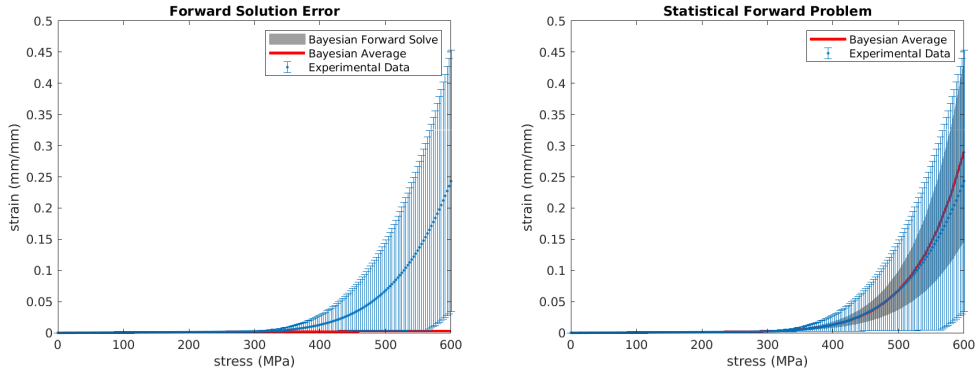

Figure B.12: Left: Updated Bayesian inference solution to LE model used for forward statistical problem. Right: Updated Bayesian inference solution to RO model used for forward statistical problem.

Both model parameters solutions shows more variation, although in case of the LE model this still results in a small range of forward solutions. The RO model does a much better job of capturing the actual data variation.

B.3.3 Model Comparison

Using the AMSSA algorithm in QUESO results in approximations of the model evidence that can be used for comparison purposes. In this example, values of model evidence were extremely large, so only model comparison is possible and not determination of model possibility.

The log of the evidence was calculated for the two models using varying amounts of data, as shown in table B.2. For low stress levels, both models should perform well - the data is linear so the LE model fits well, and the additional RO model parameters should go to zero. Indeed, using the posteriors for the forward statistical problem results in forward solutions very similar to the data. However, it can be seen that in terms of model comparison, the LE model ranks higher.

Data Level	0-20	0-100	0-200	0-300	0-400	0-500	0-600
log(LE Evidence)	17448	16977	16291	14401	12969	12104	11345
log(RO Evidence)	16526	16171	15384	14354	13179	12413	11824

Table B.2: Log of model evidence with data from increasing amounts of stress included.

This reflects the RO model's complexity - since both models fit the data well, the model with fewer parameters wins. It is only when the data becomes nonlinear (between 300-400 MPa) that the RO model begins to win. At this point, the goodness-of-fit is sufficient to balance the more shallow priors, and the posterior evidence is larger. See figure B.13 for the model comparison values.

Figure B.13: LE vs. RO model comparison.

Bibliography

[1] Martin S Alnaes. "UFL: a Finite Element Form Language". In: *Automated Solution of Differential Equations by the Finite Element Method, Volume 84 of Lecture Notes in Computational Science and Engineering*. Ed. by Anders Logg, Kent-Andre Mardal, and Garth N Wells. Springer, 2012. Chap. 17.

[2] Martin S Alnaes, Anders Logg, and Kent-Andre Mardal. "UFC: a Finite Element Code Generation Interface". In: *Automated Solution of Differential Equations by the Finite Element Method, Volume 84 of Lecture Notes in Computational Science and Engineering*. Ed. by Anders Logg, Kent-Andre Mardal, and Garth N Wells. Springer, 2012. Chap. 16.

[3] Martin S Alnaes et al. "The FEniCS Project Version 1.5". In: *Archive of Numerical Software* 3.100 (2015).

[4] Thomas S Deisboeck et al. "Multiscale cancer modeling." In: *Annual review of biomedical engineering* 13 (Aug. 2011), pp. 127–55. ISSN: 1545-4274.

[5] Kendra A. Erk, Kevin J. Henderson, and Kenneth R. Shull. "Strain Stiffening in Synthetic and Biopolymer Networks". In: *Biomacromolecules* 11.5 (May 2010), pp. 1358–1363. ISSN: 1525-7797.

[6] Kemelli C. Estacio-Hiroms and Ernesto E. Prudencio. *The QUESO Library, User's Manual*. Tech. rep. Center for Predictive Engineering, Computational Sciences (PECOS), at the Institute for Computational, and Engineering Sciences (ICES), The University of Texas at Austin, 2013.

[7] Danial Faghihi et al. *A Phase-Field Theory for Multi-constituent Diffusion and Deformation: Application to Avascular Tumor Growth*. Tech. rep. 2018.

[8] Sir Ronald Aylmer Fisher. "152: The Wave of Advance of Advantageous Genes." In: *Annals of Eugenics* 7 (1937), pp. 355–369.

[9] A. G. Gelman, G. O. Roberts, and W. R. Gilks. "Efficient Metropolis jumping rules". In: *Bayesian Statistics V*. Ed. by J.M. et al. Bernardo. 5th ed. Oxford University press, 1996, pp. 599–608.

[10] H. Haario, E. Saksman, and J. Tamminen. "An adaptive Metropolis algorithm". In: *Bernoulli* 7 (2001), pp. 223–242.

[11] H. Haario et al. "DRAM: Efficient adaptive MCMC". In: *Statistics and Computing* 16 (2006), pp. 339–354.

[12] Gabriel Helmlinger et al. "Solid stress inhibits the growth of multicellular tumor spheroids". In: *Nature Biotechnology* 15.8 (Aug. 1997), pp. 778–783. ISSN: 1087-0156.

[13] David A Hormuth et al. "A mechanically coupled reaction-diffusion model that incorporates intra-tumoural heterogeneity to predict in vivo glioma growth." In: *Journal of the Royal Society, Interface* 14.128 (Mar. 2017), p. 20161010. ISSN: 1742-5662.

[14] David A. Hormuth et al. “Mechanically Coupled Reaction-Diffusion Model to Predict Glioma Growth: Methodological Details”. In: Humana Press, New York, NY, 2018, pp. 225–241.

[15] Maarten Jaspers et al. “Ultra-responsive soft matter from strain-stiffening hydrogels.” In: *Nature communications* 5 (Dec. 2014), p. 5808. ISSN: 2041-1723.

[16] Yang Jiao and Salvatore Torquato. “Emergent Behaviors from a Cellular Automaton Model for Invasive Tumor Growth in Heterogeneous Microenvironments”. In: *PLoS Computational Biology* 7.12 (Dec. 2011). Ed. by Feilim Mac Gabhann, e1002314. ISSN: 1553-7358.

[17] Robert C Kirby. “FIAT: Numerical Construction of Finite Element Basis Functions,” in: *Automated Solution of Differential Equations by the Finite Element Method, Volume 84 of Lecture Notes in Computational Science and Engineering*. Ed. by Anders Logg, Kent-Andre Mardal, and Garth N Wells. Springer, 2012. Chap. 13.

[18] E. Konukoglu et al. “Image Guided Personalization of Reaction-Diffusion Type Tumor Growth Models Using Modified Anisotropic Eikonal Equations”. In: *IEEE Transactions on Medical Imaging* 29.1 (Jan. 2010), pp. 77–95. ISSN: 0278-0062.

[19] J. Kruschke. *Doing Bayesian Data Analysis: A Tutorial with R, JAGS, and Stan*. 2nd. Elsevier Science, 2015. ISBN: 9780124058880.

[20] E. A. B. F. Lima, J. T. Oden, and R. C. Almeida. "A hybrid ten-species phase-field model of tumor growth". In: *Mathematical Models and Methods in Applied Sciences* 24.13 (Dec. 2014), pp. 2569–2599. ISSN: 0218-2025.

[21] Anders Logg and Garth N Wells. "DOLFIN: Automated Finite Element Computing". In: *ACM Transactions on Mathematical Software* 37.2 (2010).

[22] Anders Logg, Garth N Wells, and Johan Hake. "DOLFIN: a C++/Python Finite Element Library". In: *Automated Solution of Differential Equations by the Finite Element Method, Volume 84 of Lecture Notes in Computational Science and Engineering*. Ed. by Anders Logg, Kent-Andre Mardal, and Garth N Wells. Springer, 2012. Chap. 10.

[23] Anders Logg et al. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012. ISBN: 978-3-642-23098-1.

[24] Anders Logg et al. "FFC: the FEniCS Form Compiler". In: *Automated Solution of Differential Equations by the Finite Element Method, Volume 84 of Lecture Notes in Computational Science and Engineering*. Ed. by Anders Logg, Kent-Andre Mardal, and Garth N Wells. Springer, 2012. Chap. 11.

[25] A. Mira. "On Metropolis-Hastings algorithm with delayed rejection". In: *Metron* 59 (2001a), pp. 231–241.

[26] E. Prudencio and S. Cheung. "Parallel Adaptive Multilevel Sampling Algorithms for the Bayesian Analysis of Mathematical Models". In: *International Journal for Uncertainty Quantification* 2 (2012), pp. 215–237.

[27] H. L. Rocha et al. “A hybrid three-scale model of tumor growth”. In: *Mathematical Models and Methods in Applied Sciences* 28.01 (Jan. 2018), pp. 61–93. ISSN: 0218-2025.

[28] Akulapalli Sudhakar. “History of Cancer, Ancient and Modern Treatment Methods”. In: *Journal of cancer science & therapy* 1.2 (2009), p. 1.

[29] Zhihui Wang et al. “Simulating cancer growth with multiscale agent-based modeling.” In: *Seminars in cancer biology* 30 (Feb. 2015), pp. 70–8. ISSN: 1096-3650.